arXiv:2303.12177v1 [cs.LG] 3 Mar 2023

EZtune：用于在 R 中自动进行超参数调优的软件包

Jill Lundell达纳-法伯癌症研究所数据
科学系 哈佛大学陈曾熙公共卫生学院生物
统计学系 马萨诸塞州波士顿 02215
jlundell@ds.dfci.harvard.edu

2023年3月23日

摘要

近年来，统计学习模型日益普及。这些模型中有许多包含必须经过调优才能发挥良好性能的超参数。调优这些参数并非易事。EZtune 是一个 R 软件包，提供简单易用的界面，可对支持向量机、AdaBoost、梯度提升机和弹性网络进行调优。我们首先简要介绍 EZtune 可以调优的模型，并讨论每种模型的超参数。随后，我们比较 EZtune、caret 和 tidymodels 的易用性。接着，我们比较使用 EZtune 和 tidymodels 调优的模型在准确率和计算时间方面的表现。最后，我们演示如何使用 EZtune 帮助选择预测能力最优的最终模型。我们的比较表明，EZtune 可以对支持向量机和梯度提升机进行调优；此外，EZtune 还提供了易于使用的界面，适合统计学习模型或 R 的初学者。

1 引言

统计学习模型为传统统计模型（如回归）提供了强有力的替代方案。然而，许多此类模型都包含必须经过调优才能实现最佳预测准确率的超参数。已有许多方法被提出用于调优统计学习模型的超参数，但其中只有少数方法有研究支持。流行的 R 包 [1]`tidymodels` [2] 和 `caret` [3]可以自动调优超参数，但对于经验较少的 R 用户或机器学习新手来说，这些包可能难以实现。我们介绍一个名为 `EZtune` 的软件包，它可以自动调优支持向量机（SVM） [4], 梯度提升机（GBM） [5], AdaBoost [6], 和弹性网络 [7]的超参数。 `EZtune` 具有简单易用的用户界面，即使是 R 新手也能轻松上手；它采用的超参数调优方法有详尽的文档说明，而且其持续调优出准确模型的能力也得到了研究支持 [8]。首先，我们简要介绍支持向量机、提升树和弹性网络，并重点说明各自的超参数。随后概述 `EZtune`、 `tidymodels`和 `caret`。接下来，我们比较 `EZtune` 与 `tidymodels` 和 `glmnet` [9] 在超参数调优方面的表现。没有免费午餐定理指出，没有任何一种模型类型能在所有情况下都胜过其他模型 [10]。因此，最后我们将演示如何使用 `EZtune` 调优不同的支持向量机、梯度提升机和弹性网络模型，从而选出性能最佳的模型。

2 调优参数概述

以下部分将简要概述支持向量机、提升树和弹性网络，并指出各模型的超参数。每个概述都着重指出相应模型类型中需要调优的超参数。

2.1 支持向量机

支持向量机使用分离超平面为分类和回归模型创建决策边界 [4]。分离超平面称为软间隔，因为它允许部分点落在超平面的错误一侧。代价参数 C 决定了对点落在间隔错误一侧的容忍度。 C 值较大时，允许许多点落在错误一侧；而 C 值较小时，对误分类点的容忍度则低得多。核函数 K 会将分类器映射到更高维空间。在高维空间中使用超平面进行分类，因此在原始空间中形成非线性边界。支持向量机的模型如下：

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(x, x_i; \gamma)$$

其中， K 是带有调优参数 γ 的核函数， S 是支持向量集合（位于间隔边界上的点），而 α_i 使用 C 和间隔计算得出。支持向量机分类的超参数为 C 和 γ。常见的核函数包括多项式核、径向核和线性核。 `tidymodels` 和 `caret` 会对这三种核函数进行调优，而 `EZtune` 仅针对径向核提供自动调优。不过，径向核在大多数情况下都表现良好。

支持向量回归（SVR）还有一个额外的调优参数， ϵ。SVR 试图找到一个函数或超平面，使得对于每个观测值 [11]，超平面与响应值 y_i 之间的偏差都小于 ϵ 。成本表示允许距离超平面超过 ϵ 的点的数量。本质上，支持向量机试图尽可能增加位于间隔正确一侧的点的数量，而 SVR 则试图尽可能增加落在间隔 ϵ 范围内的点的数量。支持向量机和 SVR 的超参数唯一的数学限制是它们必须大于 0。

2.2 提升树

提升树属于集成方法，它将许多弱学习器或分类器组合成一个准确的分类器。弱学习器通常单独使用时效果不佳，但将多个弱学习器组合起来可以构建一个强分类器 [12]。在提升树中，使用只有少数终端节点的回归树或分类树来构建弱学习模型。随后检查该树中的误分类点或残差，并利用这些信息拟合一棵新树。通过将新树添加到此前拟合的树中来更新模型。集成模型以这种方式迭代更新，最终预测值由弱学习器加权投票得出。

各种提升树算法之间的主要区别，在于每次迭代时从误分类观测值中学习的方法。AdaBoost 通过对训练数据中的所有观测值赋予相同的权重 [6]，拟合一棵小型决策树。随后，对误分类点赋予比正确分类点更高的权重，并计算一棵新树。新的预测值是所有先前决策树的加权预测值之和。该过程会重复多次，并不断提高误分类点的权重。利用加权数据创建一棵新树，然后将其添加到先前的模型中。弱学习器通过加权平均的方式组合，其中表现最佳的弱学习器获得最高权重。由此得到一个加法模型，其最终预测值是集成中所有模型所作预测值的加权和 [12]。

梯度提升机（GBM）是使用梯度下降在学习过程中最小化损失函数的提升树 [5]。损失函数可以根据待解决的问题进行调整。在本文的示例中，我们将均方误差（MSE）用作回归模型的损失函数，并将对数损失用作分类问题的损失函数。初始弱学习器采用决策树。梯度提升机递归地针对先前决策树的残差拟合新树，然后组合所有树的预测值，以获得最终预测。

AdaBoost 和梯度提升机的超参数几乎完全相同。这两种模型都需要调优迭代次数、树的深度和收缩率，后者控制树的学习速度。梯度提升机还有一个额外的超参数：终端节点中的最小观测值数量。

2.3 弹性网络

弹性网络是一种线性模型，它结合了 ℓ_1 和 ℓ_2 正则化。正则化通过引入一些偏差来降低模型的变异性。 ℓ_1 惩罚项会使模型变得稀疏。然而，仅使用 ℓ_1 惩罚项会将非零系数变量的数量限制在观测值数量以内，并阻止对变量进行组选择。也就是说，如果一组变量彼此相关，通常只会选择其中一个变量。引入 ℓ_2 正则化可以允许更多非零系数，并促使相关变量组保留在模型中。弹性网络使用以下方程估计系数：

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} ||\boldsymbol{y} - \boldsymbol{X}\beta||^2 + \lambda_2||\beta||_2^2 + \lambda_1||\beta||_1$$

参数 λ_1 和 λ_2 控制模型中 ℓ_1 和 ℓ_2 正则化的程度。岭回归是弹性网络的一种特殊情况，其中 $\lambda_1 = 0$。系数会向 0 收缩，但没有任何一个会等于 0，因此模型会保留所有预测变量。同样，套索回归是$\lambda_2 = 0$ 的弹性网络模型，因此许多系数会被设为 0。 λ_1 的值越大，系数收缩得越多。

弹性网络有两个超参数： α和 λ。参数 α是弹性网络调优参数，用于控制模型中 ℓ_1和 ℓ_2正则化的程度。其定义为 $\alpha = \frac{\lambda_1}{\lambda_1+\lambda_2}$。请注意， $\alpha \in [0, 1]$，其中 $\alpha = 0$ 是岭回归模型，而 $\alpha = 1$是套索回归模型。另一个调优参数 λ控制收缩程度。 λ 的值越大，收缩越多。对 λ 唯一的数学限制是 $\lambda \geq 0$。

3 可用 R 包讨论

有多个 R 软件包可用于调整统计学习模型。诸如 `e1071` [13] 这样的软件包可以调整单一类型的模型。不过，我们希望能通过简单的界面对不同类型的模型进行比较。因此，我们在此仅讨论最常用且可调整不同类型模型的 R 包： `caret`、 `tidymodels` 和 `EZtune`。

`caret` 软件包 [3] 是一款功能强大的软件包，多年来一直可在 R 中使用。 `caret` 几乎可以通过任何方法调整任何模型。不过，功能如此丰富，也使得学习 `caret` 耗时较长，对非专业 R 用户而言可能难以掌握且不易使用。本文不将 `caret` 纳入比较，尽管它应用广泛，但我们认为，使用它所需的编程和机器学习知识，使其不适合作为与 `EZtune` 比较的对象。

`tidymodels` [2] 是一套软件包，可通过不同程度的自动化来自动调整许多监督学习模型。 `tidymodels` 不如 `caret` 强大或灵活，但学习和使用起来容易得多。 `tidymodels` 可以调整许多不同类型的模型，并允许用户使用网格搜索或迭代贝叶斯优化 [14] 来调整模型。 `tidymodels` 包含自动为网格搜索选择合理范围的功能，这对不了解超参数的用户很有帮助。不过，用户仍需知道需要调整哪些超参数，以及构建不同模型时使用了哪些 R 包。虽然 `tidymodels` 比 `caret` 容易使用得多，但新手 R 用户仍难以使用；要学会使用它，仍需充分了解不同模型及其超参数。

`EZtune` [8] 调优的模型数量少于 `caret` 或 `tidymodels`，但其用户界面简单易用，适合 R 新手或缺乏机器学习模型使用经验的用户。调优通过使用 Hookes-Jeeves 算法 [15] 或遗传算法 [16]优化超参数空间来完成。 `EZtune`无需了解超参数及其特性。该界面旨在适用于计算流程或 R 函数。

4 EZtune 与其他 R 包的比较

本节比较了 `EZtune` 与 `tidymodels` 在支持向量机和梯度提升机调优方面的表现。 `tidymodels`无法调优弹性网络模型，因此我们比较了 `EZtune` 和 `glmnet` [9] 在弹性网络调优方面的表现。本节未纳入 Adaboost，因为 `tidymodels` 无法调优 Adaboost。本节旨在展示 `tidymodels` 和 `EZtune` 的优势与不足，并提供一篇关于如何

表 1：用于探索超参数的数据集列表。

数据集	N	变量	分类变量	连续变量
鲍鱼	4177	9	1	7
波士顿房价 2	506	19	1	15
CO2	84	5	3	1
犯罪	47	14	1	12
乳腺癌	699	10	0	9
Pima	768	9	0	8
声呐	208	61	0	60
地衣	840	40	2	31
毛蕊花	12094	32	0	31

注：
鲍鱼数据来自 AppliedPredictiveModeling 软件包 [17]。波士顿房价 2、乳腺癌、皮马和声呐数据来自 mlbench 软件包 [18]。二氧化碳数据来自 datasets 软件包 [1]。犯罪数据来自《Practicing Statistics》一书 [19]。地衣和毛蕊花数据为 EZtune 内部数据 [8]。

使用这两个软件包。文中包含许多代码片段，演示如何将这两个软件包用于不同类型的模型和调优方法。

比较中采用了分类和回归模型，因为不同模型和软件包在这两种情形下的表现各不相同。五个数据集具有二元响应，用于分类；另外四个数据集具有连续响应变量，用于比较回归方法。选择这些数据集是因为它们可公开获取，且曾用于以往的基准测试研究。数据集的说明见表 1。

使用 `rsample` package [20] 将数据集划分为训练集和测试集，并使用训练数据集对模型进行调优。使用划分得到的测试数据验证调优后的模型，并针对每种方法和每个数据集比较结果。以下代码展示了如何划分所有二元分类测试所用的数据。回归数据集采用相同的方法，但初始的_split 函数中不使用 strata 参数。

```
library(mlbench)
library(rsample)
data(Sonar)
sonar_split <- initial_split(Sonar, strata = Class)
sonar_train <- training(sonar_split)
sonar_test <- testing(sonar_split)
sonar_folds <- vfold_cv(sonar_train)
```

对模型进行调优，并记录十次试验中的准确率或均方根误差（RMSE）以及计算时间。报告每个数据集和调优方法的平均计算时间和平均准确率。`EZtune` 使用遗传算法和 Hooke-Jeeves 算法进行测试，并在调优时采用 10 折交叉验证和快速验证方法。快速方法将数据随机分成两半，用其中一半训练模型，再用另一半验证模型 [8]。梯度提升机和支持向量机的比较采用 `tidymodels`，并使用网格搜索和迭代贝叶斯优化。网格为每个由 `tidymodels` 选定的超参数设置五个不同取值；迭代贝叶斯优化则进行十次迭代。弹性网络使用 `glmnet` 进行调优，采用第 4.3 节所述的两种不同方法。每节均包含用于执行计算的代码示例。

4.1 支持向量机的结果

`tidymodels` 使用程序包 `kernlab` [21]，而 `EZtune` 使用程序包 `e1071` [13] 作为支持向量机计算引擎。`EZtune` 仅调整采用径向核的模型，而 `tidymodels` 可以调整采用

线性核、多项式核或径向核的模型。为确保可比性，所有比较均采用径向核。二元分类模型的成本和 γ 均进行了调整，回归模型也调整了 ϵ 。

以下代码片段展示了如何使用 `tidymodels` 通过迭代贝叶斯优化，针对 Sonar 数据调优支持向量机。通过使用 svm_rbf 函数创建模型，以指定模型为支持向量机，并确定要调优的超参数。随后配合 set_engine 指定用于支持向量机计算的底层引擎包，并使用 set_mode 定义模型类型。使用 metric_set 确定用于模型调优的性能指标。接着使用 workflow、add_model 和 add_formula 函数指定调优工作流。指定模型和工作流后，函数 tune_bayes 会使用模型参数、工作流以及性能指标集，通过迭代贝叶斯调优支持向量机。通过 final_workflow 重新拟合调优后的模型，随后使用 last_fit 和 collect_metrics 获取测试数据集上的指标，从而得到性能结果。这个工作流在各个阶段都提供了很大的灵活性，但要将各个部分拼接起来并确定每个部分的输入，可能颇具挑战。

```
library(tidymodels)

tune_model <- svm_rbf(cost = tune(), rbf_sigma = tune()) %>%
  set_engine("kernlab") %>%
  set_mode("classification")
mets <- metric_set(accuracy, roc_auc)

model_wf <- workflow() %>%
  add_model(tune_model) %>%
  add_formula(Class ~ .)

model_set <- parameters(model_wf)best_model <- model_wf %>%
tune_bayes(resamples = sonar_folds, param_info = model_set,
initial = 5,  iter = 10,  metrics = mets) %>%
select_best("accuracy")

results <- model_wf %>%
  finalize_workflow(best_model) %>%
  fit(data = sonar_train) %>%
  last_fit(sonar_split, metrics = mets) %>%
  collect_metrics()

as.data.frame(results[, c(1, 3)])
```

以下代码片段演示了如何使用 `EZtune` ，通过遗传算法和 10 折交叉验证为 Sonar 数据调优支持向量机。请注意， `EZtune` 只需调用一次 eztune 函数即可调优支持向量机，而 `tidymodels` 则需要调用十个函数才能获得调优后的模型。对于 R 新手用户而言，获取测试数据集性能指标的方法也比 `tidymodels`简单直观得多。

```
library(EZtune)

model <- eztune(x = subset(sonar_train, select = -Class),y = sonar_train$Class,
 method = "svm",  optimizer = "ga", fast = FALSE,  cross = 10)


predictions <- predict(model, sonar_test)
acc <- accuracy_vec(truth = sonar_test$Class, estimate = predictions[, 1])
auc <- roc_auc_vec(truth = sonar_test$Class, estimate = predictions[, 2])

data.frame(Accuracy = acc, AUC = auc)
```

表 2：分类支持向量机调优十次试验的平均准确率和计算时间（秒）。每个数据集的最佳准确率和最短计算时间以粗体标出。

	EZtune				tidymodels	
Data	遗传算法交叉验证	遗传算法快速	HJ交叉验证	HJ快速	Grid	IB
准确率						
乳腺癌	0.994	0.993	**0.996**	0.993	0.965	0.965
地衣	**0.900**	0.892	0.871	0.886	0.856	0.842
毛蕊花	**0.959**	0.949	**0.959**	0.957	0.884	0.916
Pima	0.833	**0.847**	0.827	0.822	0.763	0.737
声呐	0.948	**0.959**	0.954	0.957	0.814	0.882
时间（秒）						
乳腺癌	9.32	3.47	1.35	**0.591**	126	88.0
地衣	59.1	15.7	14.4	**3.74**	146	92.8
毛蕊花	47,800	3,550	38,300	**1,170**	7,380	3,310
Pima	38.2	9.26	5.91	**1.38**	122	84.3
声呐	9.54	4.61	2.76	**1.56**	111	87.7

平均准确率和平均计算时间（秒）如表 2 所示。表中显示，对于全部五个数据集，最佳准确率均由 `EZtune` 获得。此外，所有数据集的最短计算时间均由 `EZtune` 使用 Hooke-Jeeves 优化算法和快速选项实现。所有 `EZtune` 运行的计算时间都比 `tidymodels` 更短，其中一些 `EZtune` 运行甚至比 `tidymodels` 运行快 50 到 100 倍。例外情况是使用交叉验证进行调优的最大数据集毛蕊花。

在四个数据集上进行了支持向量回归。回归模型采用的方法与二元分类模型相同，不同之处在于，除了成本和 γ，还对 ϵ 进行了调优。使用 `EZtune` 进行回归的代码与二元分类的代码完全相同，因为 `EZtune` 会自动根据响应变量的类型进行相应调整。 `tidymodels` 则需要稍作修改，以指定模型是分类模型还是回归模型。与二分类支持向量机试验一样，使用训练数据集通过每种方法对模型进行调优，然后使用测试数据集验证模型。

以下 代码片段展示了如何为回归测试划分波士顿房价数据集 ts.

```
library(mlbench)
data(BostonHousing2)
bh <- mutate(BostonHousing2, lcrim = log(crim)) %>%
  dplyr::select(-town, -medv, -crim)
bh_split <- initial_split(bh)
bh_train <- training(bh_split)
bh_test <- testing(bh_split)
bh_folds <- vfold_cv(bh_train)
```

以下代码片段演示了如何使用 `tidymodels` 通过迭代贝叶斯优化对波士顿住房数据中的支持向量机进行调优。该工作流与用于二元分类支持向量机的工作流类似。主要区别在于，模型被指定为回归模型，添加了 ϵ 作为超参数，并使用均方根误差和平均绝对误差作为验证模型的指标。

```
tune_model <- svm_rbf(cost = tune(), rbf_sigma = tune(), margin = tune()) %>%
  set_engine("kernlab") %>%
  set_mode("regression")
mets <- metric_set(rmse, mae)

model_wf <- workflow() %>%
  add_model(tune_model) %>%
  add_formula(cmedv ~ .)
```

表 3：回归支持向量机调优十次试验的平均均方根误差和计算时间（以秒为单位）。每个数据集的最佳结果以粗体显示。

	EZtune				Tidymodels	
Data	遗传算法交叉验证	遗传算法快速	HJ交叉验证	HJ快速	Grid	IB
RMSE						
鲍鱼	2.16	2.15	**2.09**	2.11	2.11	2.13
波士顿住房	**2.82**	3.12	3.50	2.94	3.09	2.89
CO2	4.20	**3.80**	4.44	4.31	4.28	4.79
犯罪	26.7	28.9	**24.6**	26.8	30.2	28.0
时间（秒）						
鲍鱼	8,410	272	309	**26.5**	1,980	327
波士顿住房	91.5	4.75	41.0	**1.56**	426	91.1
CO2	6.99	2.06	1.49	**0.442**	414	137
犯罪	1.93	1.87	0.573	**0.436**	447	107

```
model_set <- parameters(model_wf)best_model <- model_wf %>%
tune_bayes(resamples = bh_folds, param_info = model_set,
initial = 5, iter = 10, metrics = mets) %>%select_best("rmse")


results <- model_wf %>%
  finalize_workflow(best_model) %>%
  fit(data = bh_train) %>%
  last_fit(bh_split, metrics = mets) %>%
  collect_metrics()

as.data.frame(results[, c(1, 3)])
```

以下代码片段演示了如何使用遗传算法和 10 折交叉验证，通过 `EZtune` 对波士顿住房数据的支持向量机进行调优。请注意，使用 eztune 的语法与二分类支持向量机相同。这是因为 eztune 会根据响应变量判断模型是分类模型还是回归模型，然后相应地调整超参数、调优区域和验证指标。

```
model <- eztune(x = subset(bh_train, select = -cmedv), y = bh_train$cmedv,
method = "svm", optimizer = "ga", fast = FALSE, cross = 10)

predictions <- predict(model, bh_test)
rmse.ez <- rmse_vec(truth = bh_test$cmedv, estimate = predictions)
mae.ez <- mae_vec(truth = bh_test$cmedv, estimate = predictions)
data.frame(RMSE = rmse.ez, MAE = mae.ez)
```

每种模型类型均运行十次，并计算 RMSE；表 3 列出了各方法在每个数据集上的平均 RMSE，以及每次运行的平均计算时间。表中显示，各方法的 RMSE 相近，但所有最低的 RMSE 均由 `EZtune` 获得。使用 Hooke-Jeeves 算法和快速选项的 `EZtune` 计算时间最短。`EZtune` 在鲍鱼数据上使用遗传算法和交叉验证时，计算时间最长。这与表 2 中二元分类结果所呈现的情况相似：在大型数据集上使用遗传算法和交叉验证，计算速度也比其他 `EZtune` 和 `tidymodels` 选项慢。遗传算法结合交叉验证所得的准确率和 RMSE 并不优于其他选项，这意味着对于较大的数据集而言，较长的计算时间可能并不值得。

4.2 梯度提升机的结果

tidymodels 使用软件包 xgboost [22]，而 EZtune 使用软件包 gbm [23] 作为梯度提升机的引擎。梯度提升机的全部四个调优参数均使用 tidymodels 和 EZtune 进行调优。与支持向量机类似，tidymodels 使用网格搜索和迭代贝叶斯算法运行，网格大小和迭代次数采用与支持向量机相同的标准。EZtune 则采用与支持向量机迭代相同的标准运行。

以下代码片段展示了如何使用网格搜索对 tidymodels 的 Sonar 数据进行梯度提升机调优。

```
tune_model <- boost_tree(trees = tune(), tree_depth = tune(),
learn_rate = tune(),min_n = tune()) %>%set_engine("xgboost") %>%
set_mode("classification")mets <- metric_set(accuracy, roc_auc)
model_wf <- workflow() %>%add_model(tune_model) %>%
add_formula(Class ~ .)best_model <- model_wf %>%
tune_grid(resamples = sonar_folds, grid = 5^4, metrics = mets) %>%
select_best("accuracy")results <- model_wf %>%
finalize_workflow(best_model) %>%fit(data = sonar_train) %>%
last_fit(sonar_split, metrics = mets) %>%collect_metrics()
as.data.frame(results[, c(1, 3)])
```

以下代码片段展示了如何使用 EZtune 结合 Hooke-Jeeves 和快速选项为 Sonar 数据调优梯度提升机。

```
model <- eztune(x = subset(sonar_train, select = -Class),
y = sonar_train$Class,  method = "gbm",  optimizer = "hjn", fast = 0.5)

predictions <- predict(model, sonar_test)
acc <- accuracy_vec(truth = sonar_test$Class, estimate = predictions[, 1])
auc <- roc_auc_vec(truth = sonar_test$Class, estimate = predictions[, 2])
data.frame(Accuracy = acc, AUC = auc)
```

表4显示了十次试验的平均准确率和平均计算时间。表中显示，EZtune 的准确率明显高于 tidymodels，乳腺癌数据的差距约为 3 个百分点，而 Sonar 数据的差距高达 14 个百分点。对于所有数据集，使用 Hooke-Jeeves 算法和快速选项的 EZtune 计算时间最短，速度约为 tidymodels 迭代贝叶斯选项的 10 倍甚至更快。Hooke-Jeeves 快速选项的准确率也与所有数据集上获得的最优准确率相近。tidymodels 的网格搜索选项比其他模型慢得多。这是因为每个超参数都测试了五个选项。使用梯度提升机进行分类的网格搜索需要进行 625 次测试，而二元分类的支持向量机调优只需 25 次。除了 Sonar 数据之外，迭代贝叶斯在 tidymodels上的表现几乎与网格搜索一样好。

以下代码演示了　　如何使用 tidymodels 对波士顿房价数据集上的梯度提升机进行调优　　，方法是使用网格搜索。

```
tune_model <- boost_tree(trees = tune(), tree_depth = tune(),
learn_rate = tune(),  min_n = tune()) %>%set_engine("xgboost") %>%
```

表4：分类梯度提升机调优十次试验的平均准确率和计算时间（秒）。每个数据集的最佳结果以粗体显示。

	EZtune				Tidymodels	
Data	遗传算法交叉验证	遗传算法快速	HJ交叉验证	HJ快速	Grid	IB
准确率						
乳腺癌	0.991	0.992	0.994	**0.995**	0.965	0.965
地衣	0.895	**0.898**	0.891	0.893	0.844	0.852
毛蕊花	**0.970**	**0.970**	0.967	0.966	0.929	0.922
Pima	**0.833**	0.823	0.806	0.815	0.742	0.740
声呐	0.924	**0.935**	0.934	0.904	0.863	0.794
时间（秒）						
乳腺癌	1,440	79.0	199	**13.9**	5,540	196
地衣	4,130	369	853	**50.3**	11,200	433
毛蕊花	149,000	7,770	24,600	**1,160**	194,000	7,970
Pima	935	66.5	210	**13.6**	5,050	208
声呐	1,730	79.5	306	**17.8**	6,170	200

```
  set_mode("regression")
mets <- metric_set(rmse, mae)

model_wf <- workflow() %>%
  add_model(tune_model) %>%
  add_formula(cmedv ~ .)

best_model <- model_wf %>%
  tune_grid(resamples = bh_folds, grid = 5^4, metrics = mets) %>%
  select_best("rmse")

results <- model_wf %>%
  finalize_workflow(best_model) %>%
  fit(data = bh_train) %>%
  last_fit(bh_split, metrics = mets) %>%
  collect_metrics()
as.data.frame(results[, c(1, 3)])
```

以下代码片段展示了如何使用 `EZtune` ，通过 Hooke-Jeeves 和快速选项对波士顿住房数据的梯度提升机进行调优。

```
model <- eztune(x = subset(bh_train, select = -cmedv), y = bh_train$cmedv,
method = "gbm", optimizer = "hjn", fast = 0.5)


predictions <- predict(model, bh_test)
rmse.ez <- rmse_vec(truth = bh_test$cmedv, estimate = predictions)
mae.ez <- mae_vec(truth = bh_test$cmedv, estimate = predictions)
data.frame(RMSE = rmse.ez, MAE = mae.ez)
```

表5显示了回归试验中均方根误差和平均计算时间。与二元分类一样，使用 `tidymodels` 的网格搜索明显慢于其他选项，且均方根误差没有显著改善。 `EZtune` 的快速计算耗时远短于其他方法，其中 Hooke-Jeeves 的计算时间最短。四个数据集中有三个的最佳均方根误差结果由 `EZtune`取得。

表5：回归梯度提升机调优十次试验的平均均方根误差和计算时间（秒）。每个数据集的最佳结果均以粗体显示。

	EZtune				tidymodels	
Data	遗传算法交叉验证	遗传算法快速	HJ交叉验证	HJ快速	Grid	IB
RMSE						
鲍鱼	2.18	2.14	2.17	2.16	**2.13**	2.15
波士顿住房	2.63	2.79	2.97	**2.48**	2.92	3.00
CO2	2.60	2.80	**2.48**	2.71	2.59	2.55
犯罪	25.8	31.2	27.5	**22.6**	24.1	31.2
时间（秒）						
鲍鱼	8,110	523	4,180	**289**	32,800	675
波士顿住房	3,180	171	1,480	**64.0**	6,840	288
CO2	98.6	6.54	49.9	**3.35**	3,380	176
犯罪	81.3	2.84	41.2	**1.20**	3,600	128

4.3 弹性网络的结果

`glmnet` [9] 可用于调优弹性网络，但无法同时调优 λ 和 α 。本文将使用 `EZtune` 进行自动调优与一种使用 `glmnet`的常见调优方法进行比较。 `glmnet` 方法如下：

1. 对每个 α（0、0.1、0.2、. . .、0.9、1.0），执行以下操作：2. 使用 cv.glmnet 找出能获得最佳准确率或均方根误差（最小值-λ）的 λ，以及产生的误差在最小值一个标准误差范围内的 λ （1-SE）。3. 选择能产生最佳准确率或均方根误差模型的 α 和 λ 组合。对每种 λ 类型（最小值-λ 和 1-SE）分别执行此操作。

与之前的比较一样，使用试验数据集调整 `EZtune` 和 `glmnet` 模型，然后使用测试数据集进行验证。请注意，`EZtune` 使用 `glmnet` 同时调整 λ 和 α，但它使用 Hooke-Jeeves 或遗传算法在超参数空间中进行搜索。

以下代码片段展示了如何使用 `glmnet`对声呐数据进行弹性网络调参。请注意， `glmnet` 对 glmnet 和 cv.glmnet 函数所用数据的格式要求十分严格。解释变量必须是矩阵，因此因子变量或字符变量不能直接用于这些函数。 `EZtune` 对传递给 eztune 函数的数据格式要求较为宽松。它既能处理数据框对象，也能处理矩阵对象，还能直接处理字符变量和因子变量。

```
library(glmnet)

foldid <- sample(1:10, size = nrow(sonar_train), replace = TRUE)
alpha <- seq(0, 1, 0.1)
alpha_data <- data.frame(alpha = alpha, lambda = NA, loss = NA)
model_cv <- NULL

for (i in 1:length(alpha)) {
model_cv[[i]] <- cv.glmnet(x = as.matrix(subset(sonar_train, select = -Class)),
y = sonar_train$Class, family = "binomial",type.measure = "class")
alpha_data[i, -1] <- c(model_cv[[i]]$lambda.1se,
model_cv[[i]]$cvm[model_cv[[i]]$lambda ==model_cv[[i]]$lambda.1se])}


model <- glmnet(x = as.matrix(subset(sonar_train, select = -Class)),
y = sonar_train$Class, family = "binomial",
lambda = alpha_data$lambda[alpha_data$loss ==min(alpha_data$loss)][1],
```

表6：十次分类弹性网络模型调优试验的平均准确率和计算时间（秒）。每个数据集的最佳结果以粗体显示。

	EZtune				glmnet	
Data	遗传算法交叉验证	遗传算法快速	HJ交叉验证	HJ快速1-SE	Min	
准确率						
乳腺癌	0.959	**0.971**	0.962	**0.971**	0.959	0.965
地衣	0.851	0.848	0.841	0.832	**0.855**	0.848
毛蕊花	0.773	0.769	**0.781**	0.772	0.776	0.778
Pima	**0.784**	0.773	0.771	0.758	0.766	0.763
声呐	0.726	0.736	0.774	**0.792**	0.717	0.745
时间（秒）						
乳腺癌	6.04	2.53	2.37	**1.45**	3.69	3.69
地衣	89.7	12.8	75.1	**9.90**	42.2	42.2
毛蕊花	1,080	143	1,440	**124**	680	680
Pima	7.94	2.91	2.55	**1.45**	2.06	2.06
声呐	33.6	5.28	5.82	**2.86**	11.3	11.3

```
    alpha = alpha_data$alpha[alpha_data$loss ==min(alpha_data$loss)][1],
type.measure = "class")
sonar_test_truth <- as.factor(as.numeric(sonar_test$Class) - 1)
result <- predict(model, as.matrix(subset(sonar_test, select = -Class)),
type = "response")result.r <- as.factor(round(result))
acc <- accuracy_vec(truth = sonar_test_truth, estimate = result.r)
auc <- roc_auc_vec(truth = sonar_test_truth, estimate = result[, 1],
event_level = "second")

data.frame(Accuracy = acc, AUC = auc)
```

以下代码片段展示了如何使用 `EZtune` 通过 Hooke-Jeeves 算法和 10 折交叉验证来调优弹性网络模型。请注意，使用 `EZtune`调优弹性网络模型要比使用 `glmnet`容易得多。

```
model <- eztune(x = subset(sonar_train, select = -Class),y = sonar_train$Class,
 method = "en",  optimizer = "hjn", fast = FALSE,  cross = 10)


predictions <- predict(model, sonar_test)
acc <- accuracy_vec(truth = sonar_test$Class, estimate = predictions[, 1])
auc <- roc_auc_vec(truth = sonar_test$Class, estimate = predictions[, 2])

data.frame(Accuracy = acc, AUC = auc)
```

表6显示了全部十次试验的平均准确率和平均计算时间。表中显示，没有一种方法在全部或大多数数据集上都取得最高准确率，而且各方法的准确率相近。使用 `EZtune` 搭配 Hooke-Jeeves 优化器和快速选项时，计算时间比使用其他选项快得多。在其中两个数据集上，这也是准确率表现最佳的选项。

以下代码片段演示如何使用`glmnet`为波士顿住房数据调优弹性网络模型。

```
bh_train$chas <- as.numeric(as.character(bh_train$chas))
bh_test$chas <- as.numeric(as.character(bh_test$chas))

foldid <- sample(1:10, size = nrow(bh_train), replace = TRUE)
alpha <- seq(0, 1, 0.1)
alpha_data <- data.frame(alpha = alpha, lambda = NA, loss = NA)
model_cv <- NULL_for (i in 1:length(alpha)) {
model_cv[[i]] <- cv.glmnet(x = as.matrix(subset(bh_train, select = -cmedv)),
y = bh_train$cmedv, family = "gaussian",type.measure = "mse")
alpha_data[i, -1] <- c(model_cv[[i]]$lambda.min,
model_cv[[i]]$cvm[model_cv[[i]]$lambda ==model_cv[[i]]$lambda.min])}
model <- glmnet(x = as.matrix(subset(bh_train, select = -cmedv)),
y = bh_train$cmedv, family = "gaussian",
lambda = alpha_data$lambda[alpha_data$loss ==min(alpha_data$loss)][1],
alpha = alpha_data$alpha[alpha_data$loss ==min(alpha_data$loss)][1],
type.measure = "mse")
result <- predict(model, as.matrix(subset(bh_test, select = -cmedv)),
type = "response")
rmse.en <- rmse_vec(truth = bh_test$cmedv, estimate = result[, 1])
mae.en <- mae_vec(truth = bh_test$cmedv, estimate = result[, 1])



data.frame(RMSE = rmse.en, MAE = mae.en)
```

以下代码片段展示了如何使用 EZtune，通过遗传算法和快速选项为波士顿住房数据调优弹性网络模型。

```
model <- eztune(x = subset(bh_train, select = -cmedv), y = bh_train$cmedv,
method = "en", optimizer = "ga", fast = 0.5)

predictions <- predict(model, bh_test)
rmse.ez <- rmse_vec(truth = bh_test$cmedv, estimate = predictions)
mae.ez <- mae_vec(truth = bh_test$cmedv, estimate = predictions)

data.frame(RMSE = rmse.ez, MAE = mae.ez)
```

表7展示了回归弹性网络模型的平均均方根误差和计算时间。与二元分类一样，没有一种方法优于其他方法。表中还显示， glmnet 方法在回归任务中的速度比在其他数据集上更快，不过所有方法都很快。

5 使用 EZtune 进行模型选择

如前所述，没有一种模型类型能在所有情况下都优于其他模型 [10]。因此，开发模型时应比较不同的模型类型。 EZtune 提供了一个简单易用的界面，可用于比较不同模型。图1展示了使用 EZtune、 tidymodels 和 glmnet 对五个二元分类数据集分别进行调优的十种模型的平均分类错误率和平均计算时间。在某些数据集上，支持向量机表现更好，而在另一些数据集上，梯度提升机表现更好。最佳模型类型还取决于所采用的模型调优方法。对于乳腺癌数据，梯度提升机和支持向量机的表现相近，其中支持向量机在大多数模型中略胜一筹。许多情况下，梯度提升机和支持向量机模型的表现相当。不过，对于某些数据集，一种模型始终优于其他模型。例如，Sonar 数据中分类错误率最低的模型是使用 EZtune 调优的支持向量机，

表7：对回归弹性网络模型进行十次调优试验的平均均方根误差和计算时间（秒）。每个数据集的最佳结果以粗体显示。

	EZtune				glmnet	
Data	遗传算法交叉验证	遗传算法快速	HJ交叉验证	HJ快速1-SE	Min	
RMSE						
鲍鱼	2.23	**2.19**	2.21	2.22	2.30	2.22
波士顿住房	4.87	4.86	4.84	4.77	4.87	**4.63**
CO2	6.07	7.34	6.53	6.64	6.51	**6.01**
犯罪	**24.4**	30.2	28.3	27.6	27.9	27.6
时间（秒）						
鲍鱼	5.71	2.67	2.20	**1.42**	1.66	1.66
波士顿住房	5.14	2.35	1.97	1.12	**0.918**	**0.918**
CO2	5.15	2.45	1.85	1.27	**0.737**	**0.737**
犯罪	5.13	2.51	1.83	1.16	**0.777**	**0.777**

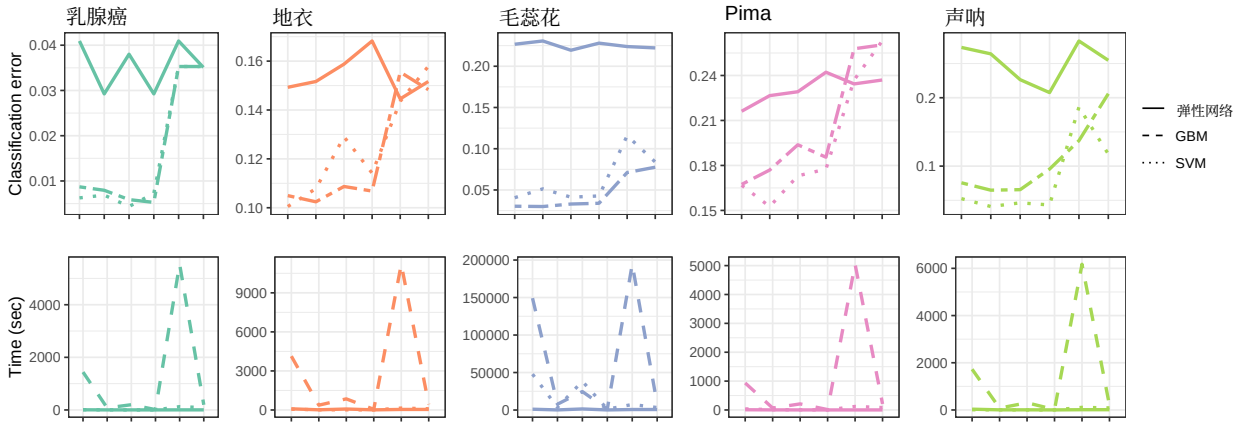


图1：二元响应数据集的分类错误率和计算时间。

而毛蕊花数据的最佳模型是使用 `EZtune` 调优的梯度提升机。模型类型（梯度提升机或支持向量机）固然重要，不同的调优方法也会产生准确率迥异的模型，地衣、皮马、鲍鱼和波士顿住房数据集便是如此。几乎在所有情况下，弹性网络模型的分类错误率都高于支持向量机和梯度提升机。

图2展示了回归模型的均方根误差和计算时间。图1和图2表明，除犯罪数据集外，对于所有数据集，弹性网络模型的错误率都高于支持向量机和梯度提升机。犯罪数据集非常小，而小型数据集的超参数调优颇具难度 [8]。支持向量机模型在鲍鱼数据上表现更好，而梯度提升机则是波士顿房价和二氧化碳数据集的更优模型。

6 结论

调优得当时，监督学习模型能够提高预测准确率，但调优这类模型并非易事。我们讨论了使用 `caret`、`tidymodels` 和 `EZtune` 调优这类模型的优缺点。这些软件包都是不错的调优选择，但各有优劣。经验丰富的 R 用户可能更青睐 `caret` 的强大功能和灵活性，但这种灵活性也意味着

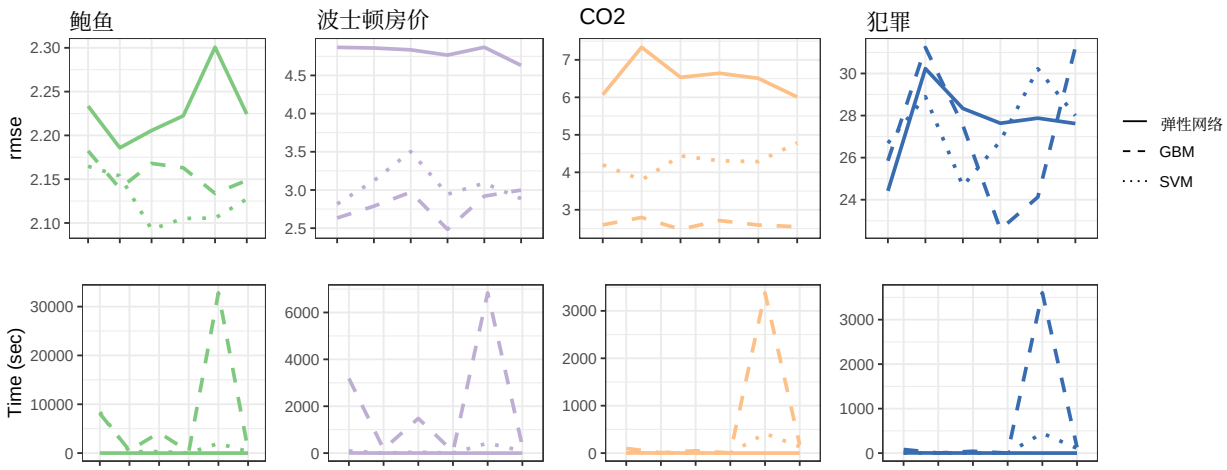


图 2：具有连续响应的数据集的均方根误差和计算时间。

即使对于经验丰富的 R 用户来说，也难以学习和实现。此外，对于深入了解超参数且希望探索更多统计学习模型的经验丰富的 R 用户而言，`tidymodels` 是不错的选择。相比之下，对于希望快速有效地对少量模型类型进行超参数调优，且不想承担其他方法所需编程工作的用户，`EZtune` 是极佳的选择。`EZtune` 是一款强大的调优工具，其简洁的界面、寻找调优良好的模型的能力以及较短的计算时间，使其成为通用超参数调优或纳入更大型计算流程的绝佳选择。`EZtune` 不仅适合刚接触统计学习模型的用户，也是一种了解统计学习模型及其超参数的绝佳方式。如果用户日后选择扩展模型范围，`EZtune` 还可帮助他们为使用 `tidymodels` 和 `caret` 做好准备。

参考文献

[1] R Core Team。《*R*：用于统计计算的语言和环境》。R 统计计算基金会，维也纳，奥地利，2022年。[2] Max Kuhn 和 Hadley Wickham。《*tidymodels*：一套基于 *tidyverse* 原则进行建模和机器学习的软件包》，2020年。[3] Max Kuhn。《*caret*：分类与回归训练》，2022年。R 软件包版本 6.0-93。[4] Corinna Cortes 和 Vladimir Vapnik。支持向量网络。《机器学习》，20(3):273–297，1995年。[5] Jerome H. Friedman。贪婪函数逼近：梯度提升机。《*Annals of Statistics*》，第 1189–1232 页，2001年。[6] Yoav Freund 和 Robert E. Schapire。在线学习的决策理论泛化及其在提升中的应用。《计算机与系统科学杂志》，55(1):119–139，1997年。[7] Hui Zou 和 Trevor Hastie。通过弹性网络进行正则化和变量选择。《英国皇家统计学会杂志：*B*辑（统计方法学）》，67(2):301–320，2005年。[8] Jill F. Lundell。《监督学习模型中的超参数调优及统计学习在全基因组关联研究中的应用：重点关注遗传力》。博士论文，犹他州立大学，2019年。[9] Jerome Friedman、Trevor Hastie 和 Robert Tibshirani。通过坐标下降求解广义线性模型的正则化路径。《*Journal of Statistical Software*》，33(1):1–22，2010年。

[10] Chris Schumacher、Michael D. Vose 和 L. Darrell Whitley。无免费午餐定理与问题描述长度。载于第三届遗传与进化计算年会论文集，第 565–570 页。Morgan Kaufmann Publishers Inc.，2001年。[11] Alex J. Smola 和 Bernhard Schölkopf。支持向量回归教程。《*Statistics and Computing*》，14(3):199–222，2004年。[12] Trevor Hastie、Robert Tibshirani 和 Jerome Friedman。《统计学习要素：数据挖掘、推断与预测》。施普林格，纽约，美国，2009年。[13] David Meyer、Evgenia Dimitriadou、Kurt Hornik、Andreas Weingessel 和 Friedrich Leisch。《*e1071*：维也纳工业大学统计系概率论研究组的杂项函数（原名：*E1071*）》，2022年。R 软件包版本 1.7-12。[14] Joao Gama。迭代贝叶斯。《*Theoretical Computer Science*》，292(2):417–430，2003年。[15] Loi Lei Lai 和 Tze Fun Chan。《分布式发电：感应发电机与永磁发电机》。约翰·威立父子出版公司，2008年。[16] D. Goldberg。《搜索优化与机器学习中的遗传算法》。Addison-Wesley Longman Publishing Company，马萨诸塞州波士顿，美国，1999年。[17] Max Kuhn 和 Kjell Johnson。《*AppliedPredictiveModeling*：应用预测建模的函数和数据集》，2018年。R 软件包版本 1.1-7。[18] David J Newman、SCLB Hettich、Cason L Blake 和 Christopher J Merz。UCI 机器学习数据库库，1998年，1998年。[19] Shonda Kuiper 和 Jeffrey Sklar。《实践统计学：第二门课程的指导性探究》。培生，马萨诸塞州波士顿，美国，2013年。[20] Hannah Frick、Fanny Chow、Max Kuhn、Michael Mahoney、Julia Silge 和 Hadley Wickham。《*rsample*：通用重采样基础设施》，2022年。R 软件包版本 1.1.1。[21] Alexandros Karatzoglou、Alex Smola 和 Kurt Hornik。《*kernlab*：基于核的机器学习实验室》，2022年。R 软件包版本 0.9-31。[22] Tianqi Chen、Tong He、Michael Benesty、Vadim Khotilovich、Yuan Tang、Hyunsu Cho、Kailong Chen、Rory Mitchell、Ignacio Cano、Tianyi Zhou、Mu Li、Junyuan Xie、Min Lin、Yifeng Geng 和 Yutian Li。*xgboost*：极端梯度提升，2022年。R 软件包版本 1.6.0.1。[23] Brandon Greenwell、Bradley Boehmke、Jay Cunningham 和 GBM 开发者。《*gbm*：广义提升回归模型》，2022年。R 软件包版本 2.1.8.1。